**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr.No** | **Table of contents** |  |
| 1  2  3  4  5  6  7  8  9 | Inspiration  Acknowledgement  Data Description  Flowchart  Libraries  Functions  Code  Output  ImportingLibraries | 2  2  3  3  3-4  4-7  7-12  13  13 |
| 10 | Importing the Dataset | 13 |
| 11 | Check Null Values | 13 |
| 12 | Unique values and Count of columns | 14 |
| 13 | Filling Null Values | 15 |
| 14 | Average Amount of Pollution In Every City | 16 |
| 15 | Most Polluted cities | 16-17 |
| 16 | Plotting graph of most polluted cities | 17-18 |
| 17 | Correlation | 18-19 |
| 18 | Graph of Pollutants in every City. | 19-22 |
| 19 | Analysis of data using Time Series | 22 |
| 20 | Plotting the Color Map for Every Month throughout the years | 23 |
| 21 | Analysis of AQI during Covid 19 Pandemic of Major Cities | 23-24 |
| 22 | Comparing AQI Before and After Lockdown using Line Graph | 24-25 |
| 23 | Creating Pivot Tables | 26 |
| 24 | Bar Plot for Comparison | 26-27 |
| 25 | Pie Plot | 31 |
| 26  27 | Box Plot  Conclusion | 32  32-34 |

**ACKNOWLEDGEMENT**

We would like to express our gratitude towards Dr. Richa Sharma and our honorable Principal Dr. Madhu Pruthi, of Keshav Mahavidyalaya for her support in accomplishment of our project on Exploratory Data Analysis on AQI Data Set.

we would like to extend our deep appreciation to all group members, without support and coordination we would not have been able to complete this project.

Group Members: Satyandra, Puneet , Saurabh.

**INSPIRATION**

Concern about quality of air we breathe. To study the different element responsible for the deteoriation of the inhaling elements. The health of the public, especially those who are the most vulnerable, such as children, the elderly and the sick, is at risk from air pollution, but it is difficult to say how large the risk is. It is possible that the problem has been over-stressed in relation to other challenges in the field of public health.

**Link to the Dataset:**

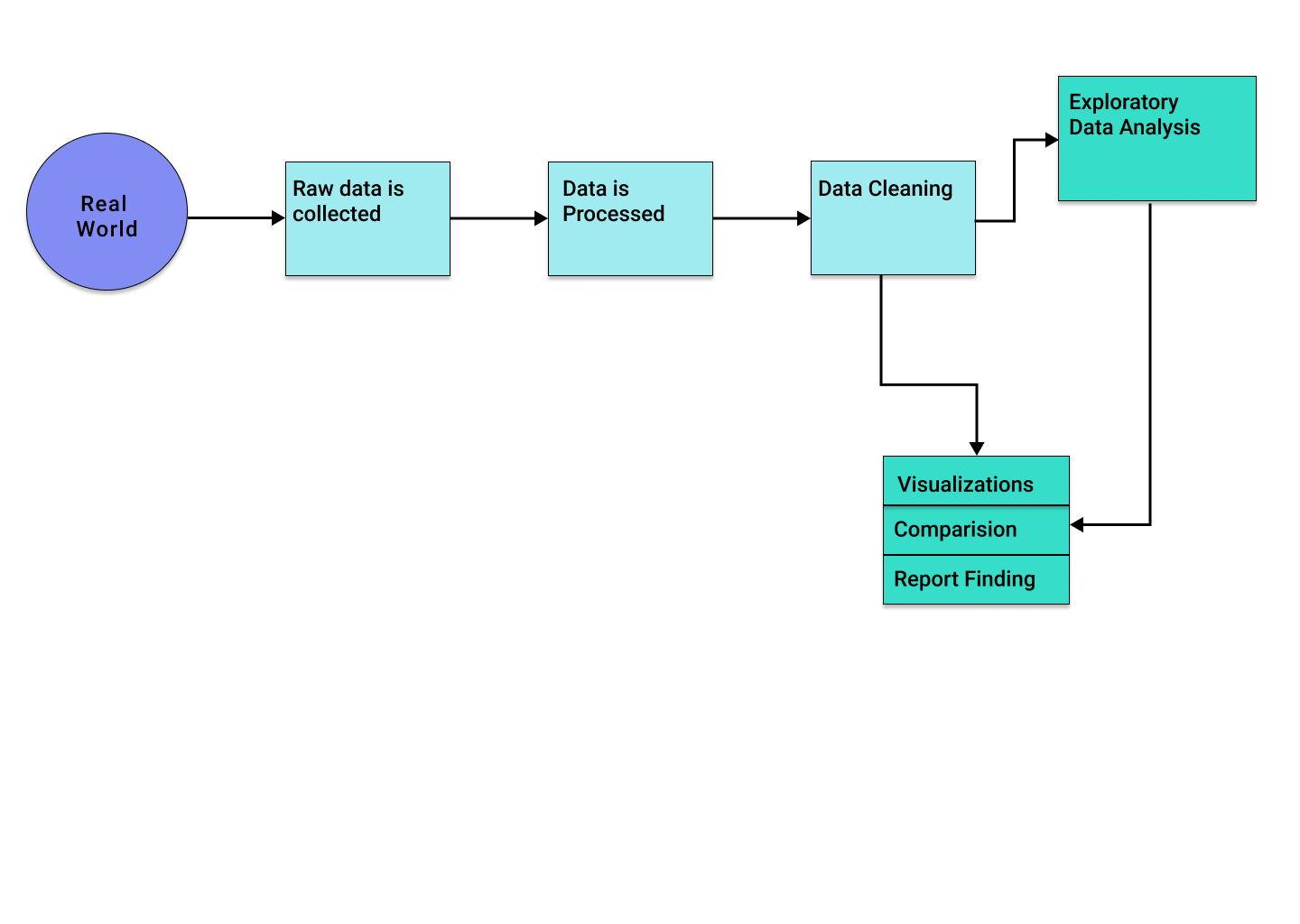
https://www.kaggle.com/rohanrao/air-quality-data-in-india

**Data Description:**

The data-set consists of 16 Columns. The columns, and their descriptions were as listed below:

1. **City**: Contains the Data of various Cities.
2. **Date**: Contains the date from 01-01-2015 – 01-07-2020.
3. **PM2.5**: Particulate Matter 2.5-Micrometer in ug/m3.
4. **PM10**: Particulate Matter 10-Micrometer in ug/m3.
5. **NO**: Nitric Oxide in ug/m3.
6. **NO2**: Nitric Dioxide in ug/m3.
7. **NOx**: Any Nitric x-oxide in ppb.
8. **NH3**: Ammonia in ug/m3.
9. **CO**: Carbon Monoxide in mg/m3.
10. **SO2**: Sulphur Dioxide in ug/m3.
11. **O3**: Ozone
12. **Benzene**: Benzene in ppb.
13. **Xylene**: Xylene in ppb.
14. **Tolune**: Tolune in ppb.
15. **AQI**: Contains the calculated value of AQI of all days
16. **AQI** **Bucket**: Contains the Category of Aqi: -
    * Good 0-50
    * Moderate 51-100
    * Satisfactory 101-150
    * Poor 151-200
    * Very Poor 201-300
    * Severe 301-Higher

**Flowchart:**

****

**LIBRARIES:**

* **NumPy** aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make #working with ndarray very easy
* **Pandas** is an open-source library that is built on top of NumPy library. It is a Python package that offers various data structures and operations for manipulating numerical data and time series. It is mainly popular for importing and analyzing data much easier.
* **Seaborn** is an open-source Python library built on top of matplotlib. It is used for data visualization and exploratory data analysis. Seaborn works easily with data frames and the Pandas library. The graphs created can also be customized easily.
* **Matplotlib** comes with a wide variety of plots. Plots helps to understand trends, patterns, and to make correlations. They’re typically instruments for reasoning about quantitative information.

**FUNCTIONS:**

1) **MERGE**

* pandas.merge(left, right, how='inner', on=None, left\_on=None, right\_on=None, left\_index=False, right\_index=False, sort=False, suffixes=('\_x', '\_y'), copy=True, indicator=False, validate=None)[source]
* Merge DataFrame or named Series objects with a database-style join.
* A named Series object is treated as a DataFrame with a single named column.
* The join is done on columns or indexes. If joining columns on columns, the DataFrame indexes will be ignored. Otherwise if joining indexes on indexes or indexes on a column or columns, the index will be passed on. When performing a cross merge, no column specifications to merge on are allowed.

Parameters:

* Left DataFrame
* right DataFrame or named Series
* Object to merge with.

**How**{‘left’, ‘right’, ‘outer’, ‘inner’, ‘cross’}, default ‘inner’

Type of merge to be performed.

* left: use only keys from left frame, similar to a SQL left outer join; preserve key order.
* right: use only keys from right frame, similar to a SQL right outer join; preserve key order.
* outer: use union of keys from both frames, similar to a SQL full outer join; sort keys lexicographically.
* inner: use intersection of keys from both frames, similar to a SQL inner join; preserve the order of the left keys.
* cross: creates the cartesian product from both frames, preserves the order of the left keys.

2) **PD.DATAFRAME**

* Pandas DataFrame is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows andcolumns).

3)**VALUES**

* values() is an inbuilt method in Python programming language that returns a list of all the values available in a given dictionary.
* Returns: returns a list of all the values available in a given dictionary. the values have been stored in a reversed manner.There are no parameters

4) **SORT\_VALUES**

* Pandas sort\_values() function sorts a data frame in Ascending or Descending order of passed Column. It’s different than the sorted Python function since it cannot sort a data frame and particular column cannot be selected.
* DataFrame.sort\_values(by, axis=0, ascending=True, inplace=False, kind=’quicksort’, na\_position=’last’)
* by: Single/List of column names to sort Data Frame by.
* axis: 0 or ‘index’ for rows and 1 or ‘columns’ for Column.
* ascending: Boolean value which sorts Data frame in ascending order if True.
* inplace: Boolean value. Makes the changes in passed data frame itself if True.
* kind: String which can have three inputs(‘quicksort’, ‘mergesort’ or ‘heapsort’) of algorithm used to sort data frame.
* na\_position: Takes two string input ‘last’ or ‘first’ to set position of Null values. Default is ‘last’.

5) **RENAME\_FUNCTION**

* rename() method in Python is used to rename a file or directory. This method renames a source file/ directory to specified
* destination file/directory. Parameters: source: A path-like object representing the file system path

6) **READ\_CSV**

* This function reads a csv file in a dataframe. The Pandas read\_csv() function returns a new DataFrame with the data and labels from the file data. csv , which you specifiedwith the first argument. This string can be any valid path, including URLs.

7)**HEAD**

* This function returns the first 5 rows.The head() function is used to get the first n rows. It is useful for quickly testing if your object has the right type of data in it. For negative values of n , the head() function returns all rows except the last n rows, equivalent to df[:-n].

8)**FIGURE(FIGSIZE=(X,Y))**

* The purpose of using plt. figure() is to create a figure object. The whole figure is regarded as the figure object. It isnecessary to explicitly use plt. figure() when we want to tweak the size of the figure and when we want to add multiple Axes objects in a single figure.
* figsize(float, float): These parameter are the width, height in inches

9) **BARPLOT**

* function calculates a summary statistic for each category
* x, y: This parameter take names of variables in data or vector data, Inputs for plotting long-form data.
* hue: (optional) This parameter take column name for colour encoding.
* data: (optional) This parameter take DataFrame, array, or list of arrays, Dataset for plotting.
* We use plt.xticks(rotation=#) where # can be any angle by which we want to rotate the x labels
* The title () function in python is the Python String Method which is used to convert the first character in each word to Uppercase and remaining characters to Lowercase in the string and returns a new string.
* plt. show () starts an event loop, looks for all currently active figure objects, and opens one or more interactive windows that display your figure or figures.

10) **INFO**

* The info() function is **used to print a concise summary of a DataFrame**.
* This method prints information about a DataFrame including the index dtype and column dtypes, non-null values and memory usage.

11) **ISNULL**

* ***Syntax:****Series.isnull()*
* ***Parameter :****None*
* ***Returns :****boolean*
* isnull() function has returned an object containing boolean values. All missing values have been mapped to True.

12) **SUM**

* **sum(iterable, start)**
* **iterable :** iterable can be anything list , tuples or dictionaries ,
* but most importantly it should be numbers.
* **start** : this start is added to the sum of
* numbers in the iterable.
* If start is not given in the syntax , it is assumed to be 0.

**Possible two syntaxes:**

* **sum(a)**

a is the list , it adds up all the numbers in the

list a and takes start to be 0, so returning

only the sum of the numbers in the list.

* **sum(a, start)**

this returns the sum of the list + start

13) **VALUE\_COUNT**

* **Index.value\_counts()** function returns object containing counts of unique values. The resulting object will be in descending order so that the first element is the most frequently-occurring element. Excludes NA values by default.
* **Syntax:** Index.value\_counts(normalize=False, sort=True, ascending=False, bins=None, dropna=True)
* **Parameters :**  
  **normalize :** If True then the object returned will contain the relative frequencies of the unique values.  
  **sort :** Sort by values  
  **ascending :** Sort in ascending order  
  **bins :** Rather than count values, group them into half-open bins, a convenience for pd.cut, only works with numeric data  
  **dropna :** Don’t include counts of NaN.
* **Returns:** counts : Series

**Code:**

Original file is located at

    https://colab.research.google.com/drive/1AxZj-xjlVCNmcjHeYlweLnBmBhXzucFA

# \*\*Importing Libraries\*\*

"""

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import datetime

#import matplotlib.dates as mdates

"""# \*\*Importing the Dataset\*\*"""

df = pd.read\_csv("AQI.csv").sort\_values(by=['City','Date'])#sorting by city and date.

df.head()

"""# \*\*Check Null Values\*\*"""

df.info() #Print a Concise Summary of Dataframe

df.isnull().sum() #sum(count) of null values.

"""# \*\*Unique values and Count of columns\*\*"""

for i in df.columns:

    print('column name:{}     unique values:{}'.format(i,len(df[i].unique())))  #Finds all the Unique Values Within the df.

cities = df['City'].value\_counts()   #Return a Series containing counts of unique values.

print(f'Total no. of different cities in the dataset : {len(cities)}')

print(cities.index)  #priting the unique values of column 'City'

"""# \*\*Filling Null Values\*\*"""

df['PM2.5']=df['PM2.5'].fillna(df['PM2.5'].mean())

df['PM10']=df['PM10'].fillna(df['PM10'].mean())

df['NO']=df['NO'].fillna(df['NO'].mean())

df['NO2']=df['NO2'].fillna(df['NO2'].mean())

df['NOx']=df['NOx'].fillna(df['NOx'].mean())

df['NH3']=df['NH3'].fillna(df['NH3'].mean())

df['CO']=df['CO'].fillna(df['CO'].mean())

df['SO2']=df['SO2'].fillna(df['SO2'].mean())

df['O3']=df['O3'].fillna(df['O3'].mean())

df['Benzene']=df['Benzene'].fillna(df['Benzene'].mean())

df['Toluene']=df['Toluene'].fillna(df['Toluene'].mean())

df['Xylene']=df['Xylene'].fillna(df['Xylene'].mean())

df['AQI']=df['AQI'].fillna(df['AQI'].mode()[0])

df['AQI\_Bucket']=df['AQI\_Bucket'].fillna('Moderate')

df.info()

"""# \*\*Average Amount Of Pollution In Every City\*\*"""

df.describe()   #Generate descriptive statistics.

df.reset\_index(drop=True,inplace=True)

df.head()

"""# \*\*Most Polluted cities\*\*"""

most\_polluted = df[['City', 'AQI', 'PM10','PM2.5', 'CO','NO', 'NO2','SO2','O3']].groupby(['City']).mean().sort\_values(by = 'AQI', ascending = False)

most\_polluted

"""## \*\*Plotting graph of most polluted cities\*\*"""

plt.style.use('seaborn-whitegrid')

f, ax\_ = plt.subplots(1,7, figsize = (20,8))

bar1=sns.barplot(x = most\_polluted.AQI, y = most\_polluted.index, palette='RdBu',ax=ax\_[0]);

bar1=sns.barplot(x = most\_polluted.sort\_values(by="PM10",ascending=False)['PM10'] , y = most\_polluted.index, palette='RdBu',ax=ax\_[1]);

bar1=sns.barplot(x = most\_polluted.sort\_values(by="PM2.5",ascending=False)['PM2.5'] , y = most\_polluted.index, palette='RdBu',ax=ax\_[2]);

bar1=sns.barplot(x = most\_polluted.sort\_values(by='CO',ascending=False)['CO'], y = most\_polluted.index, palette='RdBu',ax=ax\_[3]);

bar1=sns.barplot(x = most\_polluted.sort\_values(by='NO',ascending=False)['NO'] , y = most\_polluted.index, palette='RdBu',ax=ax\_[4]);

bar1=sns.barplot(x = most\_polluted.sort\_values(by='SO2',ascending=False)['SO2'] , y = most\_polluted.index, palette='RdBu',ax=ax\_[5]);

bar1=sns.barplot(x = most\_polluted.sort\_values(by='O3',ascending=False)['O3'] , y = most\_polluted.index, palette='RdBu',ax=ax\_[6]);

titles = ['AirQualityIndex', 'ParticulateMatter10','ParticulateMatter2.5', 'CO', 'NO', 'SO2', 'O3']

for i in range(7) :

    ax\_[i].set\_ylabel('')

    ax\_[i].set\_yticklabels(labels = ax\_[i].get\_yticklabels(),fontsize = 10);

    ax\_[i].set\_title(titles[i])

    f.tight\_layout()

"""### Correlation"""

cor = df.corr()

heatmap\_df= cor.drop(['NOx', 'NH3','O3','Toluene','Xylene', 'AQI']).drop(['NOx', 'NH3','O3','Toluene','Xylene', 'AQI'], axis=1)

f, ax = plt.subplots(figsize = (10,10))

sns.heatmap(heatmap\_df, vmax = 1, square = True, annot = True)

"""# \*\*Graph of Pollutants in every City.\*\*"""

df[['PM2.5','City']].groupby(['City']).median().sort\_values("PM2.5", ascending = False).plot.bar(color='#2C2891')

df[['PM10','City']].groupby(['City']).median().sort\_values("PM10", ascending = False).plot.bar(color='#FFB319')

df[['NO','City']].groupby(['City']).median().sort\_values("NO", ascending = False).plot.bar(color='#39A388')

df[['NO2','City']].groupby(['City']).median().sort\_values("NO2", ascending = False).plot.bar(color='#FFB830')

df[['CO','City']].groupby(['City']).median().sort\_values("CO", ascending = False).plot.bar(color='#FF2442')

df[['SO2','City']].groupby(['City']).median().sort\_values("SO2", ascending = False).plot.bar(color='#80ED99')

df[['O3','City']].groupby(['City']).median().sort\_values("O3", ascending = False).plot.bar(color='#EC9CD3')

df[['Benzene','City']].groupby(['City']).median().sort\_values("Benzene", ascending = False).plot.bar(color='#F037A5')

"""# \*\*Analaysis of data using Time Series\*\*"""

# convert column to datetime

df['Date'] = pd.to\_datetime(df['Date'])

pollutants = ['PM2.5','PM10','NO','NO2','NOx','NH3','CO','SO2','O3','Benzene','Toluene','Xylene','AQI'] #Creating a list for Pollutants

"""## Plotting the Color Map for Every Month throughout the years"""

df['month'] = pd.DatetimeIndex(df['Date']).month        #Returns Immutable ndarray-like of datetime64 data

mth\_dic = {1:'Jan',2:'Feb',3:'Mar',4:'Apr',5:'May',6:'Jun',7:'Jul',8:'Aug',9:'Sep',10:'Oct',11:'Nov',12:'Dec'} #Creating a dict of months

df['month']=df['month'].map(mth\_dic)              #Mapping of dict with index

df.groupby('month')[pollutants].mean().plot(figsize=(12,6), cmap='Spectral')

plt.legend(bbox\_to\_anchor=(1.0, 1.0))

plt.xticks(np.arange(12), mth\_dic.values())

plt.ylabel('Concentration per Cubic Meter')

"""# \*\*Analysis of AQI during Covid 19 Pandemic of Major Cities\*\*"""

cities = ['Ahmedabad','Delhi','Bengaluru','Mumbai','Hyderabad','Chennai','Lucknow']  #Major Cities

filter\_city\_date = df[df['Date'] >= '2019-01-01']

AQI = filter\_city\_date[filter\_city\_date.City.isin(cities)][['Date','City','AQI','AQI\_Bucket']] #taking values only after 2019

AQI.head()

"""# Comparing AQI Before and After Lockdown using Line Graph"""

subplot\_titles=["Bengaluru","Chennai","Delhi",'Hyderabad','Mumbai', "Ahmedabad","Lucknow"]     #Create a figure and a set of subplots

x\_line\_annotation = datetime.date(2020, 3, 25)    #Lockdown Date

f, axes = plt.subplots(7, 1, figsize=(15, 15), sharex=True)       #Sharex controls sharing of properties among x or y axes

for count, title in enumerate(subplot\_titles):

    ax = AQI[AQI['City']==title].plot(x='Date', y='AQI', kind='line', ax=axes[count], color='#161E54')

    ax.title.set\_text(title)

    ax.set\_xlim([datetime.date(2019, 1, 1), datetime.date(2020, 7, 1)])

    ax.axvline(x=x\_line\_annotation, linestyle='dashed', alpha=1, color='#FF0000')

"""From Above Line Graphs we can conclude that:

-The Value Of Aqi gradually increases during January to May

-The Value Of Aqi gradually decreases during June to September

-But After the Lockdown, the value decreses drastically over the all major cities

# \*\*Creating Pivot Tables\*\*

"""

AQI\_pivot = AQI.pivot(index='Date', columns='City', values='AQI')       #Pivot returns reshaped DataFrame

AQI\_pivot.head()

AQI\_beforeLockdown = AQI\_pivot['2020-01-01':'2020-03-25']

AQI\_afterLockdown = AQI\_pivot['2020-03-26':'2020-07-01']

df1= pd.DataFrame(AQI\_beforeLockdown.mean().reset\_index())

df2= pd.DataFrame(AQI\_afterLockdown.mean().reset\_index())

df3=df1.merge(df2, on='City')

df3 = df3.rename({'0\_x': 'BeforeLockdown', '0\_y': 'AfterLockdown'}, axis=1)

"""## \*\*Bar Plot for Comparision\*\*"""

df3.reset\_index().plot(x="City", y=["BeforeLockdown", "AfterLockdown"], kind="bar",figsize=(16,8))

plt.title("Comparision Of AQI Before and After Lockdown")

plt.xlabel("Cities")

plt.ylabel("AQI")

plt.show()

"""-The mean AQI value for Mumbai went from moderate(148.77) to satisfactory(64.35)

-The mean AQI value for Ahmedabad went from very poor(372.4) to moderate(118.8)

-The mean AQI value for Delhi went from poor(246.3) to moderate(125.27)

-The mean AQI value for Hyderabad went from moderate(94.43) to satisfactory(64.67)

-The mean AQI value for Bengaluru went from moderate(96) to satisfactory(65.4)

-The mean AQI value for Chennai went from moderate(80.31) to satisfactory(80.1)

\*\*Creating a copy of dataframe\*\*

"""

df\_cpy=df.copy()

sub\_set=df\_cpy[df\_cpy['City'].isin(['Ahmedabad','Delhi','Mumbai','Chennai','Hyderabad','Lucknow'])]

sub\_set.head()

sub\_set.groupby('City')['AQI\_Bucket'].value\_counts().to\_frame()   #Creating a dataframe

plt.figure(figsize=(20,10))

sub\_set.groupby('City')['AQI\_Bucket'].value\_counts().sort\_values(ascending=False).plot.bar(color=['#ff4000','#ff8000','#ffbf00','#ffff00','#bfff00','#80ff00','#40ff00','#00ff00','#00ff40','#00ff80','#00ffbf','#00ffff','#00bfff','#0080ff','#0040ff','#0000ff','#4000ff','#8000ff','#bf00ff','#ff00ff','#ff00bf','#ff0080','#ff0040','#ff0000','#660000','#4d0000','#330000'],edgecolor='black')

plt.show()

list=['Good','Moderate','Satisfactory','Poor','Very Poor','Severe']

plt.figure(figsize=(12,12))

plt.pie(df\_cpy['AQI\_Bucket'].value\_counts()

[0:6],labels=(df\_cpy['AQI\_Bucket'].value\_counts()

[0:6].keys()),autopct='%0.1f%%')

plt.figure(figsize=(20,10))

pollutants1 = ['NO','NO2','NOx','NH3','CO','SO2','O3','Benzene','Toluene','Xylene']

df\_cpy.boxplot(pollutants1)

# OUTPUT

# Importing Libraries

import numpy as np import pandas as pd import matplotlib.pyplot as plt import seaborn as sns import datetime

import matplotlib.dates as mdates

# Importing the Dataset

df = pd.read\_csv("AQI.csv").sort\_values(by=['City','Date'])#sorting by city and date. df.head()

**City**

**Date**

**PM2.5**

**PM10**

**NO**

**NO2**

**NOx**

**NH3**

**CO**

**SO2**

**O3**



|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | Ahmedabad | 1/1/2015 | NaN | NaN | 0.92 | 18.22 | 17.15 | NaN | 0.92 | 27.64 | 133.36 |
| **365** | Ahmedabad | 1/1/2016 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **731** | Ahmedabad | 1/1/2017 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **1096** | Ahmedabad | 1/1/2018 | 84.46 | NaN | 7.58 | 87.62 | 48.40 | NaN | 7.58 | 102.36 | 69.02 |
| **1461** | Ahmedabad | 1/1/2019 | 110.71 | NaN | 63.03 | 111.56 | 100.04 | NaN | 63.03 | 80.15 | 57.12 |

# Check Null Values

df.info() #Print a Concise Summary of Dataframe

<class 'pandas.core.frame.DataFrame'>

Int64Index: 29531 entries, 0 to 29234

Data columns (total 16 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

1. City 29531 non-null object
2. Date 29531 non-null object
3. PM2.5 24933 non-null float64
4. PM10 18391 non-null float64
5. NO 25949 non-null float64
6. NO2 25946 non-null float64
7. NOx 25346 non-null float64
8. NH3 19203 non-null float64
9. CO 27472 non-null float64
10. SO2 25677 non-null float64
11. O3 25509 non-null float64
12. Benzene 23908 non-null float64
13. Toluene 21490 non-null float64
14. Xylene 11422 non-null float64
15. AQI 24850 non-null float64 15 AQI\_Bucket 24850 non-null object dtypes: float64(13), object(3) memory usage: 3.8+ MB

df.isnull().sum() #sum(count) of null values.

City 0

Date 0

PM2.5 4598

PM10 11140

NO 3582

NO2 3585

NOx 4185

NH3 10328

CO 2059

SO2 3854

O3 4022

Benzene 5623

Toluene 8041

Xylene 18109

AQI 4681 AQI\_Bucket 4681 dtype: int64

# Unique values and Count of columns

for i in df.columns: print('column name:{} unique values:{}'.format(i,len(df[i].unique()))) #Finds all th

column name:City unique values:26 column name:Date unique values:2009 column name:PM2.5 unique values:11717 column name:PM10 unique values:12572 column name:NO unique values:5777 column name:NO2 unique values:7405 column name:NOx unique values:8157 column name:NH3 unique values:5923 column name:CO unique values:1780 column name:SO2 unique values:4762 column name:O3 unique values:7700 column name:Benzene unique values:1874 column name:Toluene unique values:3609 column name:Xylene unique values:1562 column name:AQI unique values:830 column name:AQI\_Bucket unique values:7 cities = df['City'].value\_counts() #Return a Series containing counts of unique values.

print(f'Total no. of different cities in the dataset : {len(cities)}') print(cities.index) #priting the unique values of column 'City'

Total no. of different cities in the dataset : 26

Index(['Chennai', 'Bengaluru', 'Lucknow', 'Delhi', 'Mumbai', 'Ahmedabad',

'Hyderabad', 'Patna', 'Gurugram', 'Visakhapatnam', 'Amritsar',

'Jorapokhar', 'Jaipur', 'Thiruvananthapuram', 'Amaravati',

'Brajrajnagar', 'Talcher', 'Kolkata', 'Guwahati', 'Coimbatore',

'Shillong', 'Chandigarh', 'Bhopal', 'Kochi', 'Ernakulam', 'Aizawl'], dtype='object')

# Filling Null Values

df['PM2.5']=df['PM2.5'].fillna(df['PM2.5'].mean()) df['PM10']=df['PM10'].fillna(df['PM10'].mean()) df['NO']=df['NO'].fillna(df['NO'].mean()) df['NO2']=df['NO2'].fillna(df['NO2'].mean()) df['NOx']=df['NOx'].fillna(df['NOx'].mean()) df['NH3']=df['NH3'].fillna(df['NH3'].mean()) df['CO']=df['CO'].fillna(df['CO'].mean()) df['SO2']=df['SO2'].fillna(df['SO2'].mean()) df['O3']=df['O3'].fillna(df['O3'].mean()) df['Benzene']=df['Benzene'].fillna(df['Benzene'].mean()) df['Toluene']=df['Toluene'].fillna(df['Toluene'].mean()) df['Xylene']=df['Xylene'].fillna(df['Xylene'].mean()) df['AQI']=df['AQI'].fillna(df['AQI'].mode()[0]) df['AQI\_Bucket']=df['AQI\_Bucket'].fillna('Moderate')

df.info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 29531 entries, 0 to 29234

Data columns (total 16 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

1. City 29531 non-null object
2. Date 29531 non-null object
3. PM2.5 29531 non-null float64
4. PM10 29531 non-null float64
5. NO 29531 non-null float64
6. NO2 29531 non-null float64
7. NOx 29531 non-null float64
8. NH3 29531 non-null float64
9. CO 29531 non-null float64
10. SO2 29531 non-null float64
11. O3 29531 non-null float64
12. Benzene 29531 non-null float64
13. Toluene 29531 non-null float64
14. Xylene 29531 non-null float64
15. AQI 29531 non-null float64 15 AQI\_Bucket 29531 non-null object dtypes: float64(13), object(3) memory usage: 3.8+ MB

# Average Amount Of Pollution In Every City

df.describe() #Generate descriptive statistics.

**PM2.5 PM10 NO NO2 NOx NH3**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **count** | 29531.000000 | 29531.000000 | 29531.00000 | 29531.000000 | 29531.000000 | 29531.000000 |
| **mean** | 67.450578 | 118.127103 | 17.57473 | 28.560659 | 32.309123 | 23.483476 |
| **std** | 59.414476 | 71.500953 | 21.35922 | 22.941051 | 29.317936 | 20.711370 |
| **min** | 0.040000 | 0.010000 | 0.02000 | 0.010000 | 0.000000 | 0.010000 |
| **25%** | 32.150000 | 79.315000 | 6.21000 | 12.980000 | 14.670000 | 12.040000 |
| **50%** | 58.030000 | 118.127103 | 11.53000 | 25.240000 | 27.550000 | 23.483476 |
| **75%** | 72.450000 | 118.127103 | 17.57473 | 34.665000 | 36.015000 | 23.483476 |
| **max** | 949.990000 | 1000.000000 | 390.68000 | 362.210000 | 467.630000 | 352.890000 |
| df.reset\_index(drop=True,inplace=True) df.head() | | |
| **City Date PM2.5** | | | **PM10** | **NO** | **NO2** | **NOx NH3** |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | Ahmedabad | 1/1/2015 | 67.450578 | 118.127103 | 0.92000 | 18.220000 | 17.150000 | 23.483476 |
| **1** | Ahmedabad | 1/1/2016 | 67.450578 | 118.127103 | 17.57473 | 28.560659 | 32.309123 | 23.483476 |
| **2** | Ahmedabad | 1/1/2017 | 67.450578 | 118.127103 | 17.57473 | 28.560659 | 32.309123 | 23.483476 |
| **3** | Ahmedabad | 1/1/2018 | 84.460000 | 118.127103 | 7.58000 | 87.620000 | 48.400000 | 23.483476 |
| **4** | Ahmedabad | 1/1/2019 | 110.710000 | 118.127103 | 63.03000 | 111.560000 | 100.040000 | 23.483476 |

# Most Polluted cities

most\_polluted = df[['City', 'AQI', 'PM10','PM2.5', 'CO','NO', 'NO2','SO2','O3']].groupby(['Ci most\_polluted

**AQI PM10 PM2.5 CO NO NO2**

## City

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Ahmedabad** | 334.485814 | 117.409318 | 67.728234 | 16.147420 | 20.956815 | 49.805675 | 4 |
| **Delhi** | 258.703833 | 228.413747 | 117.146631 | 1.976053 | 38.964280 | 50.763057 | 1 |
| **Lucknow** | 211.276755 | 118.127103 | 107.568277 | 2.131976 | 15.261301 | 33.188450 | 1 |
| **Patna** | 210.979010 | 119.013316 | 113.815353 | 1.591700 | 30.283315 | 36.507494 | 2 |
| **Gurugram** | 208.550923 | 150.467320 | 112.549731 | 1.321857 | 17.537607 | 23.797951 |  |
| **Talcher** | 155.490811 | 156.552639 | 62.607920 | 1.911862 | 28.071044 | 17.343337 | 2 |
| **Jorapokhar** | 139.759624 | 142.240508 | 66.406088 | 1.358846 | 12.530220 | 13.781598 | 2 |
| **Guwahati** | 139.579681 | 116.604900 | 63.692929 | 0.738284 | 20.038456 | 13.598607 | 1 |
| **Brajrajnagar** | 138.699360 | 123.094114 | 64.726798 | 1.870288 | 17.372515 | 19.524152 | 1 |
| **Kolkata** | 137.723587 | 115.798256 | 64.571464 | 0.799251 | 26.550753 | 40.032711 |  |
| **Jaipur** | 133.110413 | 123.416193 | 54.640204 | 0.809991 | 14.675238 | 32.370143 | 1 |
| **Bhopal** | 131.653979 | 119.287038 | 50.601160 | 0.923001 | 7.365372 | 31.258602 |  |
| **Amritsar** | 118.526618 | 115.353495 | 56.724459 | 0.656948 | 18.640090 | 18.883865 |  |
| **Visakhapatnam** | 114.230506 | 107.916796 | 50.191650 | 0.777069 | 13.480745 | 35.728846 | 1 |
| **Chennai** | 113.724739 | 109.815308 | 51.417112 | 1.082048 | 9.340802 | 17.067334 |  |
| **Hyderabad** | 108.754736 | 96.567339 | 48.205721 | 0.594912 | 7.953391 | 28.389182 |  |
| **Kochi** | 104.228395 | 67.335432 | 31.428519 | 1.296667 | 71.102528 | 15.449963 | 1 |
| **Mumbai** | 103.293181 | 110.006396 | 54.864359 | 0.589271 | 22.705652 | 27.429689 | 1 |
| **Chandigarh** | 96.588816 | 85.656546 | 42.428943 | 0.631349 | 10.594503 | 11.834088 | 1 |
| **Amaravati** | 96.074658 | 78.777456 | 39.614400 | 0.793211 | 5.195931 | 22.545012 | 1 |
| **Bengaluru** | 94.696864 | 89.494244 | 38.118529 | 1.840878 | 9.433523 | 27.996732 |  |
| **Ernakulam** | 92.895062 | 50.058879 | 25.994274 | 1.643175 | 23.045302 | 12.157631 |  |
| **Thiruvananthapuram** | 77.287770 | 55.093885 | 29.525274 | 0.970043 | 3.848603 | 9.914812 |  |
| **Coimbatore** | 76.176166 | 39.435543 | 29.945349 | 0.959419 | 8.830156 | 28.778153 |  |
| **Shillong** | 70.122581 | 59.099258 | 38.385603 | 0.453528 | 4.087416 | 7.663125 |  |
| **Aizawl** | 35.955752 | 24.191567 | 18.020630 | 0.283628 | 9.408053 | 0.388496 |  |

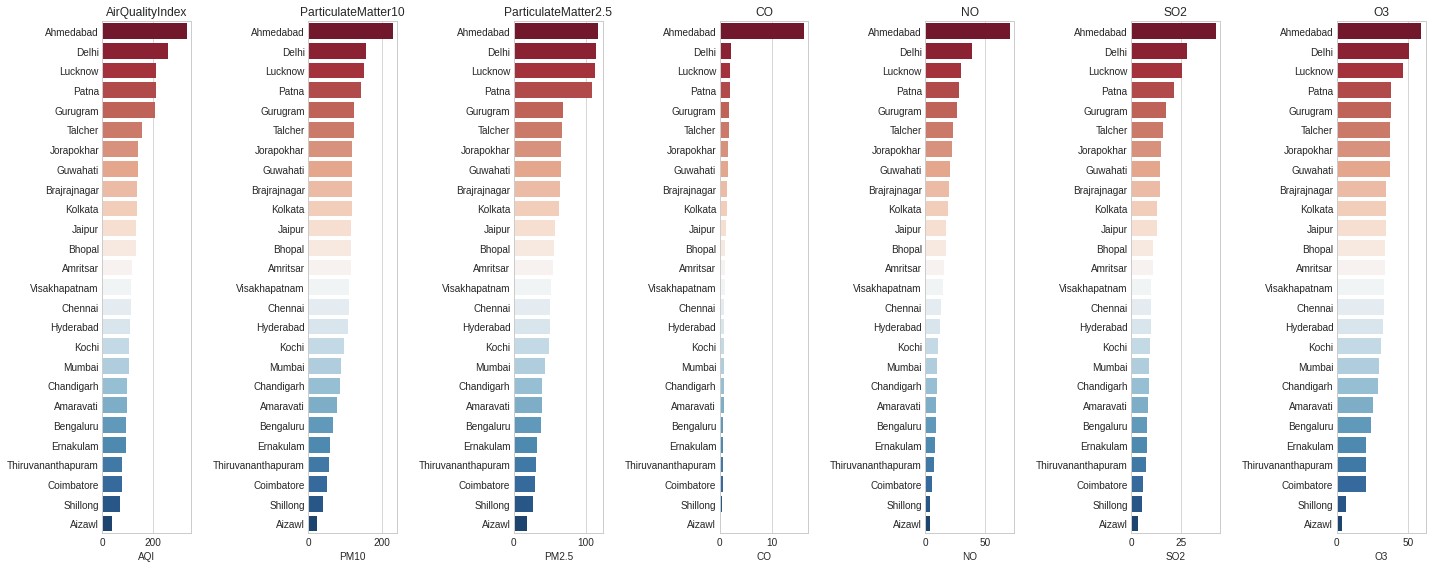
## Plotting graph of most polluted cities

plt.style.use('seaborn-whitegrid') f, ax\_ = plt.subplots(1,7, figsize = (20,8)) bar1=sns.barplot(x = most\_polluted.AQI, y = most\_polluted.index, palette='RdBu',ax=ax\_[0]); bar1=sns.barplot(x = most\_polluted.sort\_values(by="PM10",ascending=False)['PM10'] , y = most\_ bar1=sns.barplot(x = most\_polluted.sort\_values(by="PM2.5",ascending=False)['PM2.5'] , y = mos bar1=sns.barplot(x = most\_polluted.sort\_values(by='CO',ascending=False)['CO'], y = most\_pollu bar1=sns.barplot(x = most\_polluted.sort\_values(by='NO',ascending=False)['NO'] , y = most\_poll bar1=sns.barplot(x = most\_polluted.sort\_values(by='SO2',ascending=False)['SO2'] , y = most\_po bar1=sns.barplot(x = most\_polluted.sort\_values(by='O3',ascending=False)['O3'] , y = most\_poll

titles = ['AirQualityIndex', 'ParticulateMatter10','ParticulateMatter2.5', 'CO', 'NO', 'SO2', for i in range(7) :

ax\_[i].set\_ylabel('') ax\_[i].set\_yticklabels(labels = ax\_[i].get\_yticklabels(),fontsize = 10); ax\_[i].set\_title(titles[i])

f.tight\_layout()



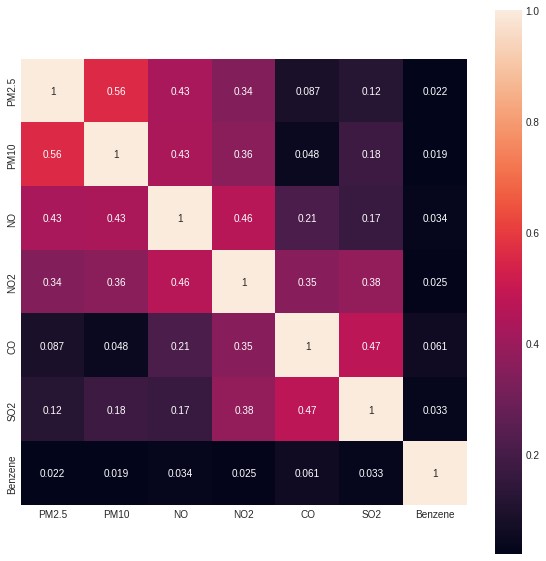
### Correlation

cor = df.corr() heatmap\_df= cor.drop(['NOx', 'NH3','O3','Toluene','Xylene', 'AQI']).drop(['NOx', 'NH3','O3',' f, ax = plt.subplots(figsize = (10,10))

h t (h t df 1 T t T )

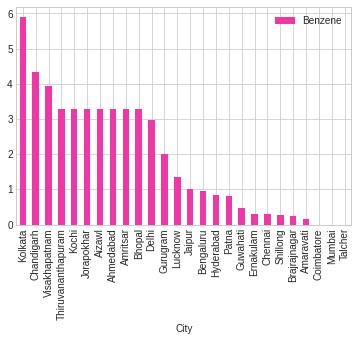
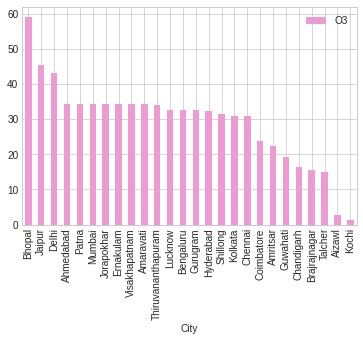
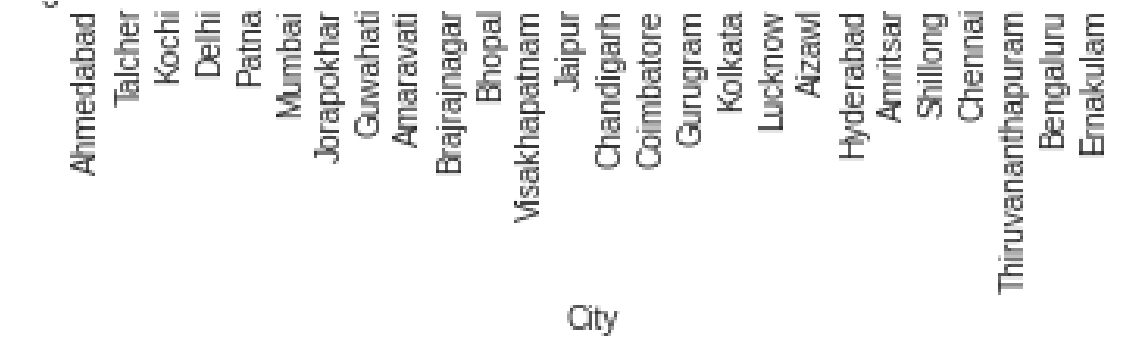
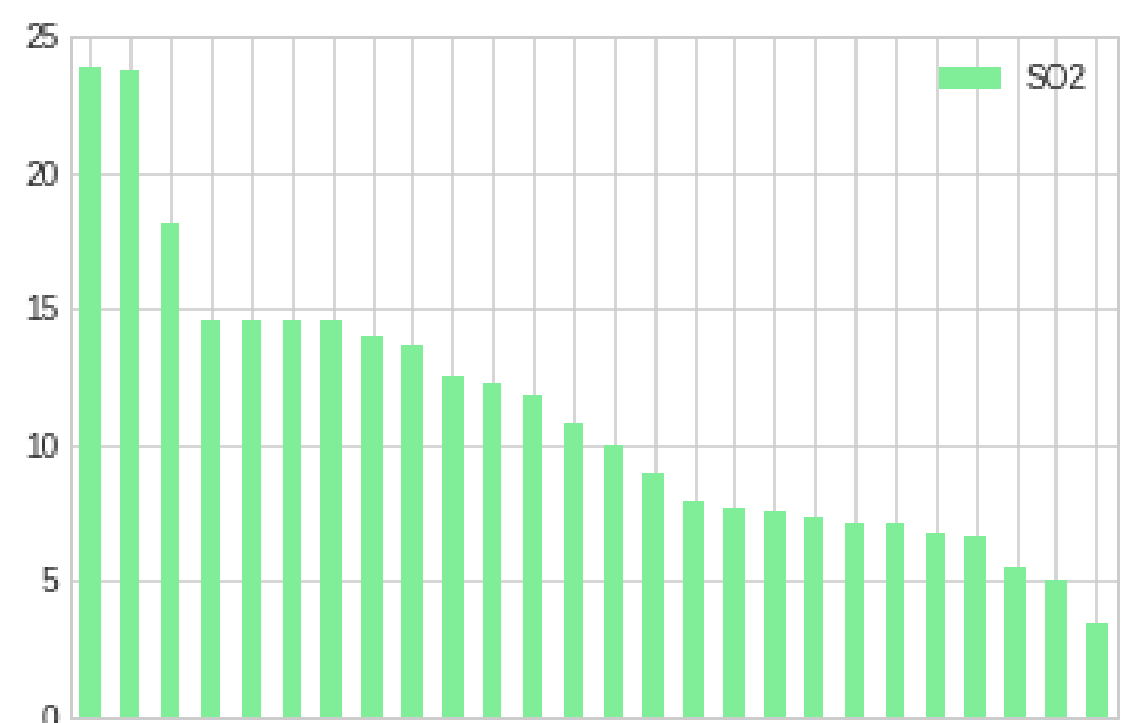
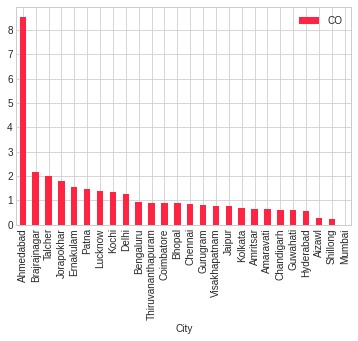
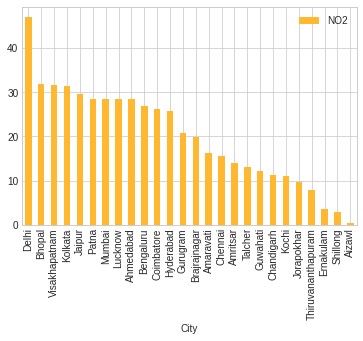
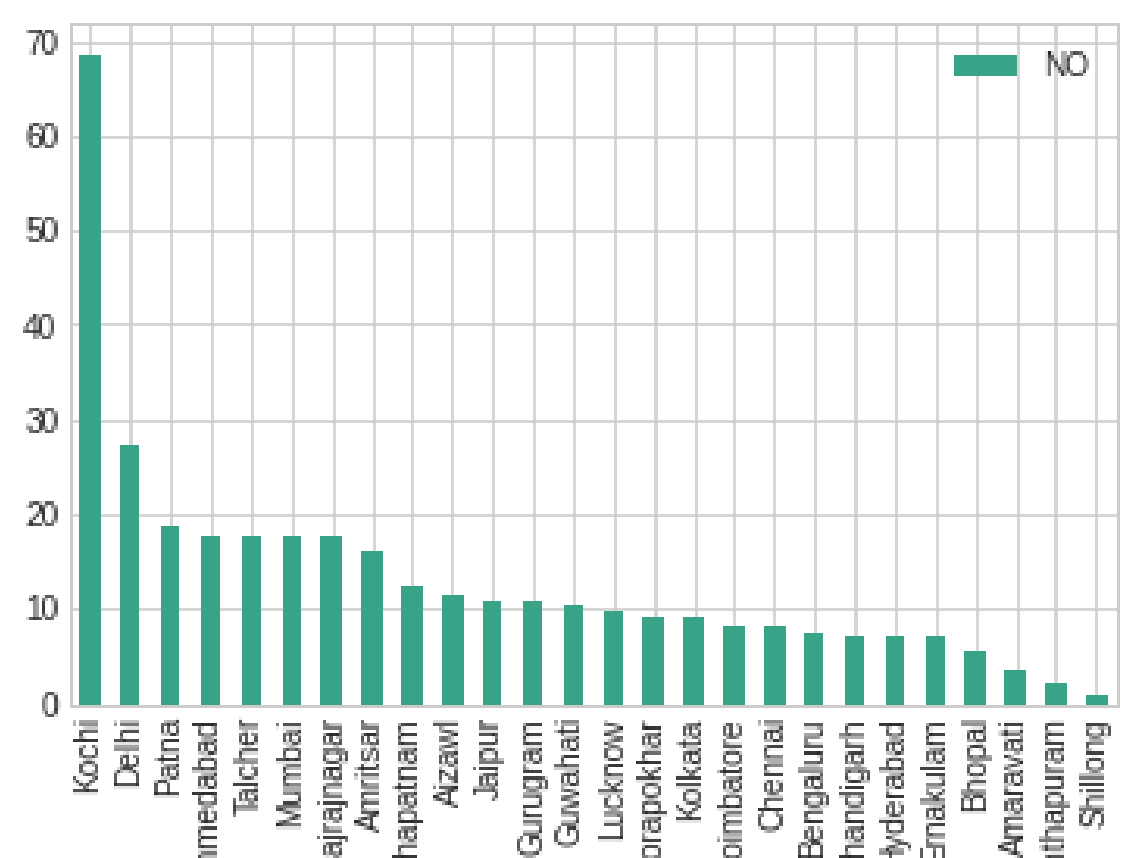
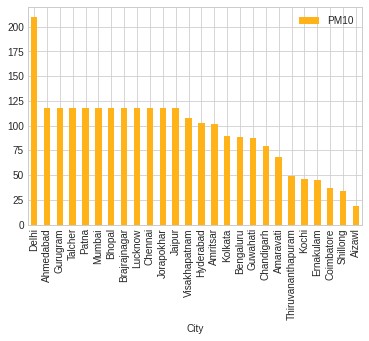
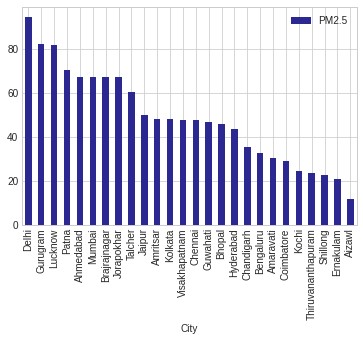
sns.heatmap(heatmap\_df, vmax = 1, square = True, annot = True)

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fde9928fc90>



**Graph of Pollutants in every City.**

df[['PM2.5','City']].groupby(['City']).median().sort\_values("PM2.5", ascending = False).plot. df[['PM10','City']].groupby(['City']).median().sort\_values("PM10", ascending = False).plot.ba df[['NO','City']].groupby(['City']).median().sort\_values("NO", ascending = False).plot.bar(co df[['NO2','City']].groupby(['City']).median().sort\_values("NO2", ascending = False).plot.bar( df[['CO','City']].groupby(['City']).median().sort\_values("CO", ascending = False).plot.bar(co df[['SO2','City']].groupby(['City']).median().sort\_values("SO2", ascending = False).plot.bar( df[['O3','City']].groupby(['City']).median().sort\_values("O3", ascending = False).plot.bar(co df[['Benzene','City']].groupby(['City']).median().sort\_values("Benzene", ascending = False).p <matplotlib.axes.\_subplots.AxesSubplot at 0x7fde9bda2350>



**Analaysis of data using Time Series**

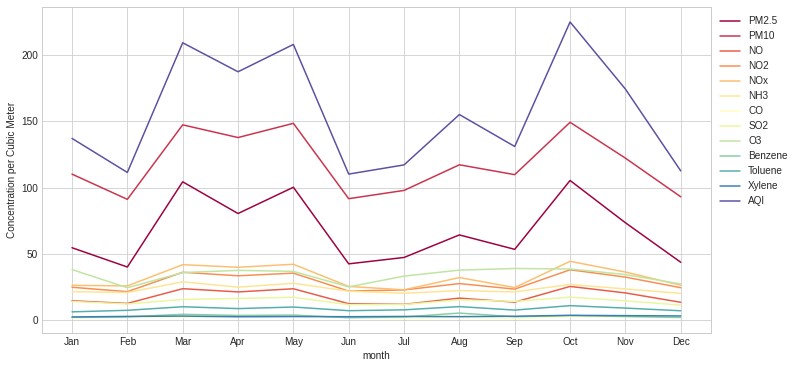
# convert column to datetime df['Date'] = pd.to\_datetime(df['Date'])

pollutants = ['PM2.5','PM10','NO','NO2','NOx','NH3','CO','SO2','O3','Benzene','Toluene','Xyle

### Plotting the Color Map for Every Month throughout the years

df['month'] = pd.DatetimeIndex(df['Date']).month #Returns Immutable ndarray-like of da mth\_dic = {1:'Jan',2:'Feb',3:'Mar',4:'Apr',5:'May',6:'Jun',7:'Jul',8:'Aug',9:'Sep',10:'Oct',1 df['month']=df['month'].map(mth\_dic) #Mapping of dict with index df.groupby('month')[pollutants].mean().plot(figsize=(12,6), cmap='Spectral') plt.legend(bbox\_to\_anchor=(1.0, 1.0)) plt.xticks(np.arange(12), mth\_dic.values()) plt.ylabel('Concentration per Cubic Meter')

Text(0, 0.5, 'Concentration per Cubic Meter')



# Analysis of AQI during Covid 19 Pandemic of Major Cities

cities = ['Ahmedabad','Delhi','Bengaluru','Mumbai','Hyderabad','Chennai','Lucknow'] #Major C filter\_city\_date = df[df['Date'] >= '2019-01-01']

AQI = filter\_city\_date[filter\_city\_date.City.isin(cities)][['Date','City','AQI','AQI\_Bucket'] AQI.head()

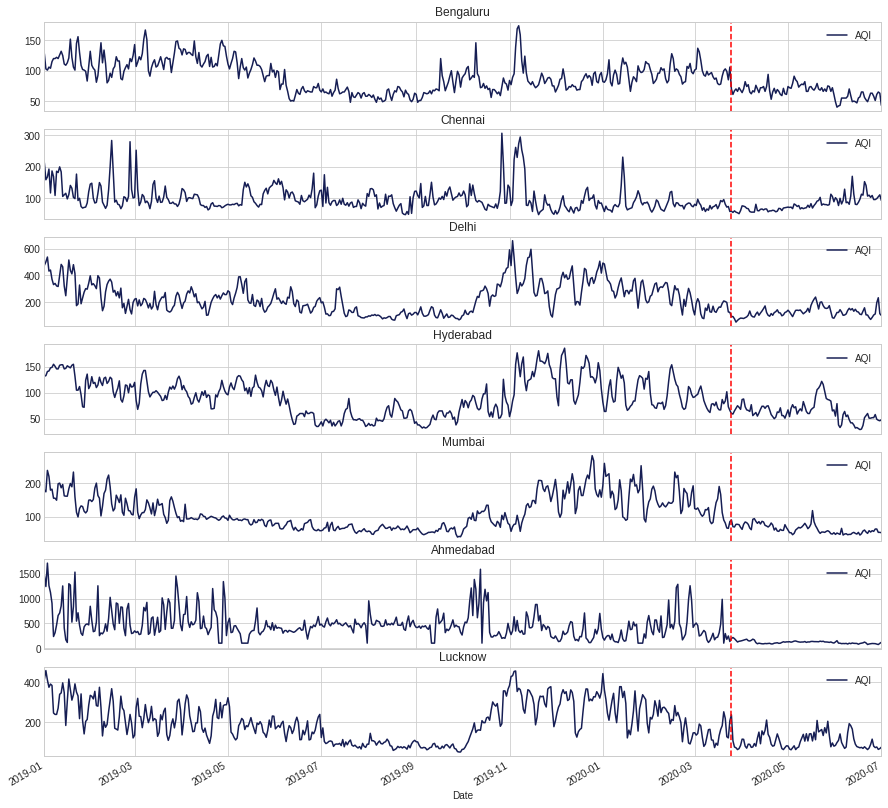
## Date City AQI AQI\_Bucket

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **4** | 2019-01-01 | Ahmedabad | 1474.0 | Severe |
| **5** | 2020-01-01 | Ahmedabad | 216.0 | Poor |
| **10** | 2019-01-10 | Ahmedabad | 661.0 | Severe |
| **11** | 2020-01-10 | Ahmedabad | 129.0 | Moderate |
| **16** | 2019-01-11 | Ahmedabad | 711.0 | Severe |

## **Comparing AQI Before and After Lockdown using Line Graph**

subplot\_titles=["Bengaluru","Chennai","Delhi",'Hyderabad','Mumbai', "Ahmedabad","Lucknow"] x\_line\_annotation = datetime.date(2020, 3, 25) #Lockdown Date f, axes = plt.subplots(7, 1, figsize=(15, 15), sharex=True) #Sharex controls sharing of for count, title in enumerate(subplot\_titles):

ax = AQI[AQI['City']==title].plot(x='Date', y='AQI', kind='line', ax=axes[count], color=' ax.title.set\_text(title) ax.set\_xlim([datetime.date(2019, 1, 1), datetime.date(2020, 7, 1)]) ax.axvline(x=x\_line\_annotation, linestyle='dashed', alpha=1, color='#FF0000')



From Above Line Graphs we can conclude that:

-The Value Of Aqi gradually increases during January to May

-The Value Of Aqi gradually decreases during June to September

-But After the Lockdown, the value decreses drastically over the all major cities

**Creating Pivot Tables**

AQI\_pivot = AQI.pivot(index='Date', columns='City', values='AQI') #Pivot returns reshaped DataFrame

| **City** | **Ahmedabad** | **Bengaluru** | **Chennai** | **Delhi** | **Hyderabad** | **Lucknow** | **Mumbai** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Date** |  |  |  |  |  |  |  |
| **2019-01-01** | 1474.0 | 128.0 | 212.0 | 475.0 | 132.0 | 413.0 | 181.0 |
| **2019-01-02** | 1246.0 | 103.0 | 158.0 | 501.0 | 133.0 | 457.0 | 175.0 |
| **2019-01-03** | 1719.0 | 101.0 | 167.0 | 537.0 | 141.0 | 410.0 | 239.0 |
| **2019-01-04** | 1264.0 | 106.0 | 192.0 | 432.0 | 142.0 | 374.0 | 221.0 |
| **2019-01-05** | 1127.0 | 104.0 | 116.0 | 440.0 | 148.0 | 390.0 | 180.0 |

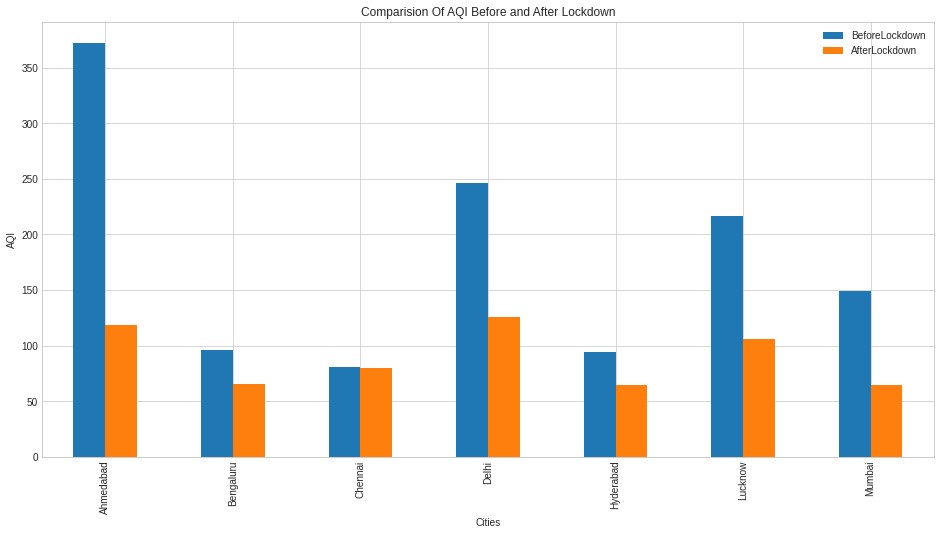
AQI\_beforeLockdown = AQI\_pivot['2020-01-01':'2020-03-25'] AQI\_afterLockdown = AQI\_pivot['2020-03-26':'2020-07-01'] df1= pd.DataFrame(AQI\_beforeLockdown.mean().reset\_index()) df2= pd.DataFrame(AQI\_afterLockdown.mean().reset\_index()) df3=df1.merge(df2, on='City')

df3 = df3.rename({'0\_x': 'BeforeLockdown', '0\_y': 'AfterLockdown'}, axis=1)

### **Bar Plot for Comparision**

df3.reset\_index().plot(x="City", y=["BeforeLockdown", "AfterLockdown"], kind="bar",figsize=(1 plt.title("Comparision Of AQI Before and After Lockdown") plt.xlabel("Cities") plt.ylabel("AQI") plt.show()

plt.show()



-The mean AQI value for Mumbai went from moderate(148.77) to satisfactory(64.35)

-The mean AQI value for Ahmedabad went from very poor(372.4) to moderate(118.8)

-The mean AQI value for Delhi went from poor(246.3) to moderate(125.27)

-The mean AQI value for Hyderabad went from moderate(94.43) to satisfactory(64.67)

-The mean AQI value for Bengaluru went from moderate(96) to satisfactory(65.4)

-The mean AQI value for Chennai went from moderate(80.31) to satisfactory(80.1) **Creating a copy of dataframe**

df\_cpy=df.copy() sub\_set=df\_cpy[df\_cpy['City'].isin(['Ahmedabad','Delhi','Mumbai','Chennai','Hyderabad','Luckn

sub\_set.head()

**City Date PM2.5 PM10 NO NO2 NOx NH3**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** Ahmedabad | 2015-  01-01 | 67.450578 | 118.127103 | 0.92000 | 18.220000 | 17.150000 | 23.483476 |
| **1** Ahmedabad | 2016-  01-01 | 67.450578 | 118.127103 | 17.57473 | 28.560659 | 32.309123 | 23.483476 |
| **2** Ahmedabad | 2017-  01-01 | 67.450578 | 118.127103 | 17.57473 | 28.560659 | 32.309123 | 23.483476 |
| **3** Ahmedabad | 2018-  01-01 | 84.460000 | 118.127103 | 7.58000 | 87.620000 | 48.400000 | 23.483476 |
| **4** Ahmedabad | 2019-  01-01 | 110.710000 | 118.127103 | 63.03000 | 111.560000 | 100.040000 | 23.483476 |

sub\_set.groupby('City')['AQI\_Bucket'].value\_counts().to\_frame() #Creating a dataframe

#### AQI\_Bucket City AQI\_Bucket

|  |  |  |
| --- | --- | --- |
| **Ahmedabad** | **Moderate** | 873 |
|  | **Severe** | 638 |
|  | **Poor** | 238 |
|  | **Very Poor** | 216 |
|  | **Satisfactory** | 43 |
|  | **Good** | 1 |
| **Chennai** | **Satisfactory** | 941 |
|  | **Moderate** | 929 |
|  | **Poor** | 110 |
|  | **Good** | 12 |
|  | **Very Poor** | 11 |
|  | **Severe** | 6 |
| **Delhi** | **Poor** | 542 |
|  | **Moderate** | 529 |
|  | **Very Poor** | 520 |
| **Severe** plt.figure(figsize=(20,10)) | | 239 |

sub\_set.groupby('City'**Satisfactory**)['AQI\_Bucket'].value\_counts().sort\_values(ascending158 =False).plot.bar(co plt.show()

|  |  |  |
| --- | --- | --- |
|  | **Good** | 21 |
| **Hyderabad** | **Moderate** | 1084 |
|  | **Satisfactory** | 710 |
|  | **Good** | 141 |
|  | **Poor** | 54 |
|  | **Very Poor** | 10 |
|  | **Severe** | 7 |
| **Lucknow** | **Moderate** | 694 |
|  | **Very Poor** | 473 |
|  | **Satisfactory** | 365 |
|  | **Poor** | 352 |
|  | **Severe** | 110 |
|  | **G d** | 15 |

**Good** 15

**Mumbai**

**Moderate**

1519

**Satisfactory**

428

**Poor**

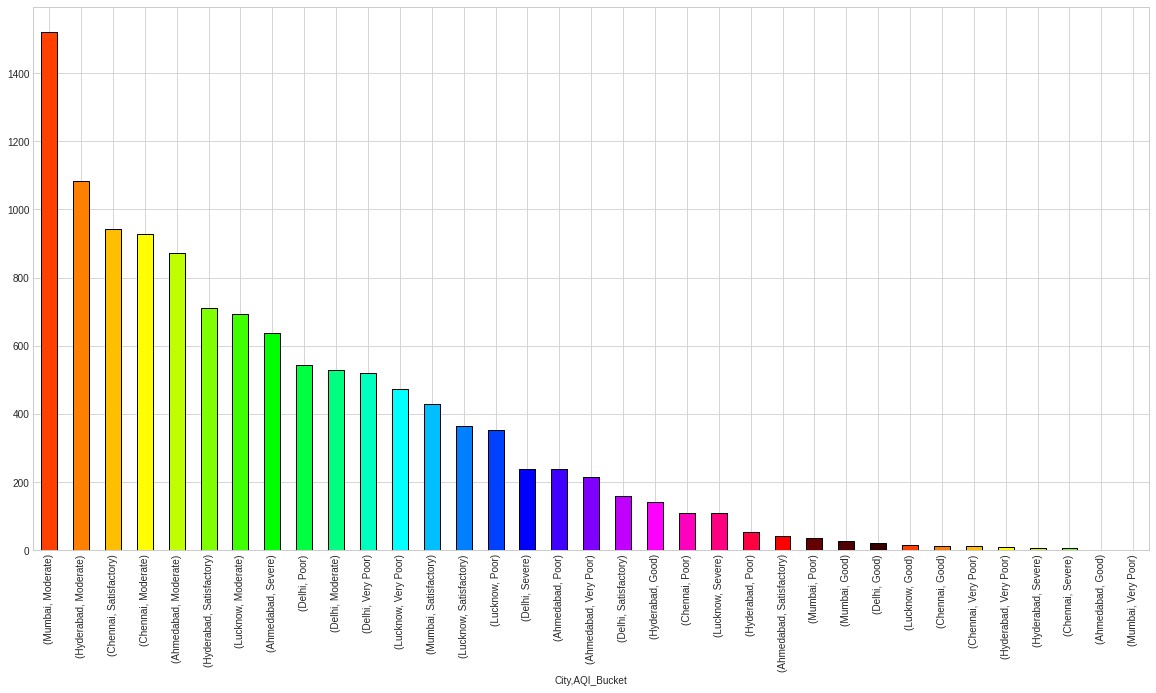
35

**Good**

26

**Very Poor**

1



list

=[

'Good'

,

'Moderate'

,

'Satisfactory'

,

'Poor'

,

'Very Poor'

,

'Severe'

]

plt.figure(figsize=(12,12)) plt.pie(df\_cpy['AQI\_Bucket'].value\_counts()

[0:6],labels=(df\_cpy['AQI\_Bucket'].value\_counts()

[0:6].keys()),autopct='%0.1f%%') ([<matplotlib.patches.Wedge at 0x7fdea4c6f3d0>,

<matplotlib.patches.Wedge at 0x7fdea4c6f790>,

<matplotlib.patches.Wedge at 0x7fdea48b1410>,

<matplotlib.patches.Wedge at 0x7fdea4e26310>,

<matplotlib.patches.Wedge at 0x7fdea4973990>,

<matplotlib.patches.Wedge at 0x7fdea49afb10>],

[Text(0.14648365529584692, 1.090202980518384, 'Moderate'),

Text(-0.9030187349109592, -0.6281378546145829, 'Satisfactory'),

Text(0.22684442834538193, -1.0763557057630424, 'Poor'),

Text(0.7515555545124892, -0.8032211703394184, 'Very Poor'),

Text(1.0010785557733146, -0.4558966167573687, 'Good'),

Text(1.0888753883698952, -0.15604611050042272, 'Severe')],

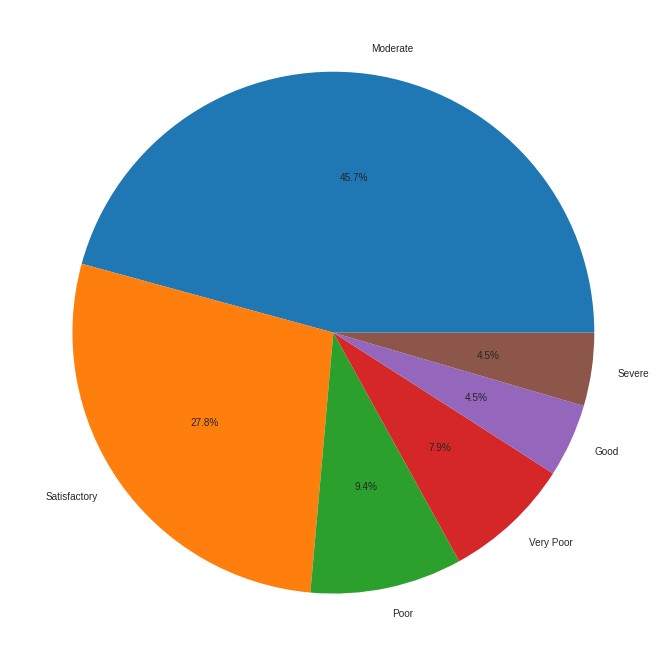
[Text(0.0799001756159165, 0.5946561711918457, '45.7%'),

Text(-0.49255567358779584, -0.3426206479715907, '27.8%'),

Text(0.1237333245520265, -0.5871031122343867, '9.4%'),

Text(0.4099393933704486, -0.4381206383669555, '7.9%'),

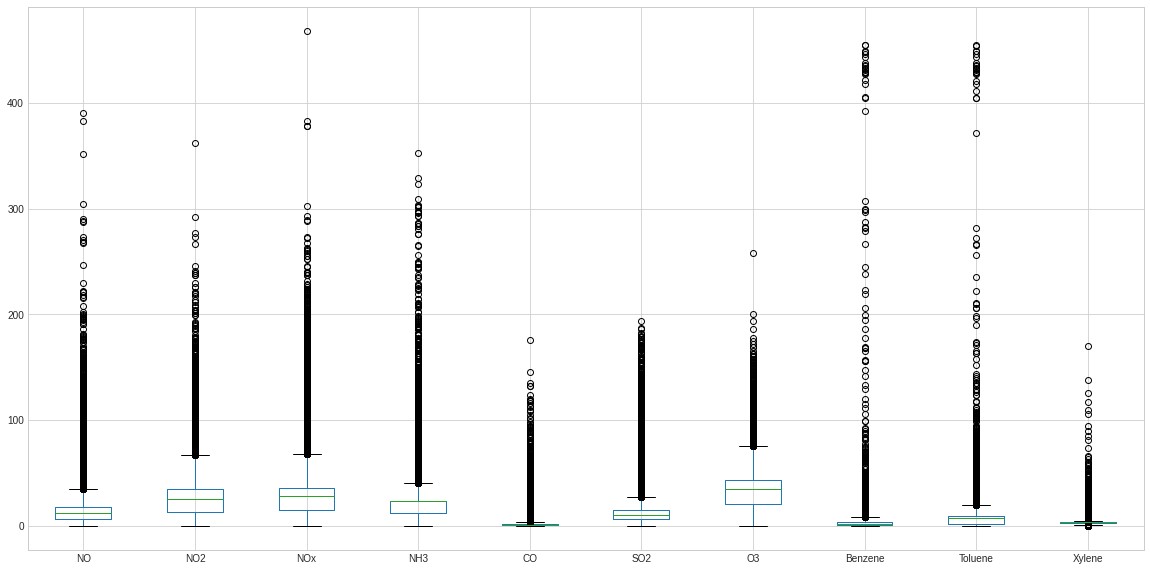
Text(0.5460428486036261, -0.24867088186765562, '4.5%'), Text(0.5939320300199428, -0.08511606027295783, '4.5%')])



plt.figure(figsize=(20,10))

pollutants1 = ['NO','NO2','NOx','NH3','CO','SO2','O3','Benzene','Toluene','Xylene']

df\_cpy.boxplot(pollutants1)



**Inferences:**

* 1. Most polluted City is Ahmedabad with AQi of 334.48.
  2. On the basis of analysis, the city with lowest AQI is Aizwal.
  3. Followed by Shillong and other North-Eastern State, that conclude that North-Eastern States have better Quality Of Air than any other metro/urban areas.
  4. AQI majorly depends on PM2.5 and PM10, CO, NO, SO3.
  5. Delhi has highest value of PM2.5, PM10 and NO2 over the period of this dataset which is 1-1-2015 – 1-7-2019.
  6. Ahmedabad has highest concentration of SO2 and CO in the air among other 26 cities. Burning of fossil fuels in the industries could be the main reason behind emission of S02 and CO in the air.
  7. Talcher is at second place in terms of SO2 Concentration and also has placed among top 5 in Concentration of CO, NO, PM10. This is because Talcher is highest coal reserve of India.
  8. With the help of Time-Series and graph Plotting Utilities, Before Lockdown it can be said that the value of AQI Gradually increases during January to May in metro cities during the period of January 2019 and March 2020, and decreases during June to September, But after the Lockdown the value of AQI decreases drastically.
  9. The more accurate data is listed below:-
     + -The mean AQI value for Mumbai went from moderate(148.77) to satisfactory(64.35)
     + -The mean AQI value for Ahmedabad went from very poor(372.4) to moderate(118.8)
     + -The mean AQI value for Delhi went from poor(246.3) to moderate(125.27)
     + -The mean AQI value for Hyderabad went from moderate(94.43) to satisfactory(64.67)
     + -The mean AQI value for Bengaluru went from moderate(96) to satisfactory(65.4)
     + -The mean AQI value for Chennai went from moderate(80.31) to satisfactory(80.1)
  10. On the basis of pie plot-
      + The AQI of around 45.7% values over each day are Moderate.
      + The AQI of around 27.8% values over each day are Satisfactory.
      + The AQI of around 9.4% values over each day are Poor.
      + The AQI of around 7.9% values over each day are Very Poor.
      + The AQI of around 4.5% values over each day are Good.
      + The AQI of around 4.5% values over each day are Severe.

**Future Scope:**

* This data can be further processed in machine learning model.
* Visualization can be done on the basis of week days vs weekends.
* Each City can be visualized separately.
* More reasons for the sudden rise and fall of the level of AQI and other pollutants could be given if we do further EDA on the dataset.

**Conclusion:**

1. This EDA gave us the idea of the rate of increase of pollution in India.
2. Different measures to lower the air pollution in India by lowering emission of fossil fuels to emit less carbon dioxide. by shifting to electric vehicle rather than fossil fuels (petrol, diesel) based vehicle.
3. The health of the public, especially those who are the most vulnerable, such as children, the elderly and the sick, is at risk from air pollution, but it is difficult to say how large the risk is. It is possible that the problem has been over-stressed in relation to other challenges in the field of public health.
4. As we have seen, there are considerable uncertainties in estimating both exposures and effects and their relationships. It may be, for example, that the effects of long-term exposure to lower concentrations of air pollutants could be more damaging to public health than short-term exposure to higher concentrations. For this reason alone, local authorities could take action to assess and improve local air quality. It is not sufficient to wait for an episode of severe air pollution and then try to deal with its effects.
5. On an individual level, the risk to health from air pollution is very much smaller than that posed by active cigarette smoking or accidents. It is also true that healthy individuals are rather unlikely to be affected by exposure to the concentrations of outdoor air pollutants in many European countries on most days of the year. However, the old and the young, and especially those suffering from respiratory or heart diseases, are the groups who are most vulnerable to the effects of air pollution. It is only right that cost effective action should be taken to provide them with clean air, which The Times of 1881 described as "the first necessity of our existence.
6. Another reason for action on air pollution is that we do not know the contribution which exposure to air pollutants may make to deaths from, for example, heart disease. In many countries heart disease is a leading cause of death and even a small contribution from air pollution could mean a significant and important effect on public health.

**References:**

1. [Air Quality Data in India (2015 - 2020) | Kaggle](https://www.kaggle.com/rohanrao/air-quality-data-in-india)
2. [National Air Quality Index (AQI) launched by the Environment Minister AQI is a huge initiative under ‘Swachh Bharat’ (pib.gov.in)](https://pib.gov.in/newsite/printrelease.aspx?relid=110654)
3. <https://www.thehindu.com/society/where-paddy-fields-sink-houses-cave-in-life-in-indias-largest-coalfield/article26744915.ece#:~:text=Black%20diamond-,Talcher%20coalfield%20has%20the%20country's%20highest%20coal%20reserve%20of%2051.163,underground%20and%20open%2Dcast%20mining>.
4. [Our Documentation | Python.org](https://www.python.org/doc/)
5. McKinney, W. (2017). Python for Data Analysis: Data Wrangling with Pandas, NumPy and I Python. 2nd edition. O’Reilly Media.